МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Лабораторная работа №4
по курсу «Программирование графических процессоров»

Работа с матрицами. Метод Гаусса.

Выполнил: И.Т. Батяновский
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

Цель работы: Использование объединения запросов к глобальной памяти.
Реализация метода Гаусса с выбором главного элемента по столбцу. Ознакомление с библиотекой алгоритмов для параллельных расчетов Thrust.

Вариант 3. Вычисление обратной матрицы.

Программное и аппаратное обеспечение

Размер глобальной памяти: 4294967296
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 65536
Максимум потоков на блок: 1024
Количество мультипроцессоров : 13
OS: Windows 10
Редактор: CLion

Метод решения

Итеративно делаем прямой ход по методу Гаусса. Для этого используем библиотеку Thrust для нахождения столбца с наибольшим элементом. Меняем эти столбцы в матрицы местами и делем нижележащие строки. Прямой ход делается на GPU, а обратный на CPU.

Описание программы

Вначале каждой итерации вычисляется адрес на нужную нам строку. И затем с той же помощью библиотеки thrust находим столбец с максимальным главным элементом. Вычитая Адрес максимального элемента с адресом строки получаем нужный индекс.

```
begin_p = thrust::device_pointer_cast(device_data + i * n);
max_p = thrust::max_element(begin_p + i, begin_p + n, comp);
ind = max_p - begin_p;
```

Если нужно менять местами меняем.

```
    if (ind != i) {
        swapRows<<<512, 512>>>(device_data, ind, i, n);
        ind = i;
```

```
}
```

И наконец делаем итерацию в прямом ходе м. Гаусса.

```
forward<<<dim3(32, 32), dim3(32, 32)>>>(device_data, i, n);
```

Реализация ядра.

```
__global__ void forward(double* data, long int i, long int n) {

	long int idx = blockIdx.x * blockDim.x + threadIdx.x;
	long int idy = blockIdx.y * blockDim.y + threadIdx.y;
	long int offsetX = gridDim.x * blockDim.x;
	long int offsetY = gridDim.y * blockDim.y;

	for (unsigned long int j = idx + i + 1; j < n; j += offsetX) {
		for (unsigned long int k = idy + i + 1; k < n + 1; k += offsetY) {
			data[j + k * n] -= data[j + i * n] * data[i + k * n] / data[i + i * n];
		}
	}
}
```

Выводы

В данной лабораторной работе я встретил снова знакомый мне с курса численных метод Гаусса для решения алгоритм LU разложения с перестановками, однако мне пришлось написать его реализацию в непривычном, но эффективном виде на GPU. Реализованный мной алгоритм является одним из самых эффективных методов для не итерационного решения СЛАУ.

Помимо этого я познакомился с тем, как эффективно обращаться к глобальной памяти с использованием объединения запросов, а также узнал про то, как пользоваться библиотекой thrust, которая предоставляет возможность эффективных и безопасных вычислений.

В ходе выполнения столкнулся с массой проблем, однако практически все они были следствием моей невнимательности и несмотря на все трудности я выполнил работу, которая наглядно показала мне, насколько большое преимущество может дать использование графического процессора вместо центрального в задачах работы с матрицами.